# Semantic Analysis on Twitter Profile 'Elon Musk'

Sheheryar Ali Bhatti - 150912

March 6, 2019

## Introduction

Twitter is an online social networking service that enables users to send and read short messages called *'Tweets'*. It has more than 300 million active users. In this document we will try to interpret the sentiments of technology entrepreneur and engineer Elon Musk from twitter using the Twitter Data API.

## Accessing Twitter API

In order to get access to the twitter data firstly you must have a twitter account. Using that twitter account you have to create an app and then twitter will provide you with four credentials which are then required to access the API. We will send these credentials to *setup_twitter_oauth* method of OAuth library to authenitcate our requests.

```
setup_twitter_oauth(API_key,API_Secret,access_token,access_token_secret)

## [1] "Using direct authentication"
```

Now we have successfully authenticated with Twitter Data API and can access the tweets.

## Retrieving Tweets

### 1) Tweets

We want to do sentiment analysis on Elon Musk and try to interpret the sentiments. Since we want the tweets for a particular profile we will make use of *userTimeline* function and pass or profile name as first argument and no. of tweets as second argument.

```
tweets<-userTimeline("elonmusk",n=500)
```

This returns the tweets in the form of list, as we want to perform different operation on this data we will convert this into data frame which will make easier for us to perform different operations.

```
tweets.df <- twListToDF(tweets)

tweets.df[1, c("id", "created", "screenName", "replyToSN", "favoriteCount",
"retweetCount", "longitude", "latitude", "text")]
```

```
##                    id             created screenName replyToSN
## 1 1103344096993660928 2019-03-06 17:18:26   elonmusk   TheRock
##   favoriteCount retweetCount longitude latitude
## 1         21307         1094        NA       NA
##                                                         text
## 1 @TheRock Oh stop, you�ll make me blush <U+263A><U+FE0F>
```

```
writeLines(strwrap(tweets.df$text[1], 60))
```

```
## @TheRock Oh stop, you�ll make me blush <U+263A><U+FE0F>
```

## Data Preprocessing

Data preprocessing is very important step.To perform any kind of algorithm fisrtly data must be clean in order to create better observations.

Extract only the text of the tweets which is available in the *text* column of the dataset.

```
data<-tweets.df$text
```

### 1) Corpus

We are dealing with text data so we will create a corpus of or tweets. A corpus or text corpus is a large and structured set of texts. Firstly we will vectorize or data i.e it will create a document for each tweet and then pass these document to the *Corpus* function.

```
myCorpus <- Corpus(VectorSource(data))
```

### 2) Removing Numbers

Just like *apply* function for list we have *tm_map* function to apply a specific function on each document.

Numbers donot have any effect on the sentiment of the document so we will remove the numbers from each document.

```
myCorpus <- tm_map(myCorpus, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, removeNumbers): transformation
## drops documents
```

### 3) Removing Url

Some tweets also consist of URLs, these urls are also not required for the sentiment analysis, so we will also remove these. We will write a function in which if the text of the document matches the Regex for Url we will remove that part of the document.

```
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeURL))
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, content_transformer(removeURL)):
## transformation drops documents
```

## 3) Removing Punctuations

Punctions are added to the sentence to make it easy for readers to read the document and understand, but they donot have any meaning. So we will remove punctuations, similarly we will create a Regex for punctuation and remove the part of the documnet which matches the regex.

```
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeNumPunct))

## Warning in tm_map.SimpleCorpus(myCorpus,
## content_transformer(removeNumPunct)): transformation drops documents
```

## 4) Lowercase

To make ur data consistent we will lower case the complete document.

```
myCorpus <- tm_map(myCorpus, content_transformer(tolower))

## Warning in tm_map.SimpleCorpus(myCorpus, content_transformer(tolower)):
## transformation drops documents
```

## 5) Stopwords

Stopwords area use to build the structure of a sentance, they have no meaning. Thus we will remove all the stopword from each document. Since some of the words maybe considered as stop word but are of great meaning in or document forexample: So will exclude these words and remove all other stop words from each document.

```
myStopwords <- c(setdiff(stopwords('english'), c("tech", "ai")), "data")
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)

## Warning in tm_map.SimpleCorpus(myCorpus, removeWords, myStopwords):
## transformation drops documents
```

## 6) WhiteSpace

Some document may have some of unnecesory whitespace and tab spaces, we will remove them from the document.

```
myCorpus <- tm_map(myCorpus, stripWhitespace)

## Warning in tm_map.SimpleCorpus(myCorpus, stripWhitespace): transformation
## drops documents
```

## 7) Diplaying Clean Data

After doing all of the preprocessing we will create a copy of or corpus and lets have a look and see how our documents look after applying preprocessing.

```
writeLines(strwrap(myCorpus[[1]]$content, 60))

## therock oh stop youll make blush
```

## Word Frequency

Word frequency gives us the better understanding of what words are common among tweets and which word used how many times in a tweet. We will create a function which takes a corpus as first argument and the word (for which you want to know the frequency) as second argument and returns the frequency of the word.

## 1) Frequency Count

```
wordFreq <- function(corpus, word) {
results <- lapply(corpus,
function(x) { grep(as.character(x), pattern=paste0("nn<",word)) })
sum(unlist(results))
}
```

## 2) Replacing Words

Some people use abbreviations and short notation for words, we will create function that replaces those jargons with full form of the word.

```
replaceWord <- function(corpus, oldword, newword) {
tm_map(corpus, content_transformer(gsub),
pattern=oldword, replacement=newword)
}
```

## 3) Term Document Matrix

Term Document Matrix is another useful technique to get great insights about our data. It is a matrix as documnet i.e tweets in our case as rows and word as columns and a numerc value for each cell which represents the frequency of the word in the document.

```
tdm <- TermDocumentMatrix(myCorpus)
inspect(tdm[1:50, 1:3])

## <<TermDocumentMatrix (terms: 50, documents: 3)>>
## Non-/sparse entries: 10/140
## Sparsity           : 93%
## Maximal term length: 14
## Weighting          : term frequency (tf)
## Sample             :
##           Docs
## Terms      1 2 3
##   blush    1 0 0
##   lift     0 0 1
##   little   0 0 1
##   make     1 0 0
##   plumazul 0 1 0
##   stop     1 0 0
##   tesla    0 1 0
##   therock  1 0 0
##   yeah     0 0 1
##   youll    1 0 0
```

```
idx <- which(dimnames(tdm)$Terms %in% c("tesla", "ai", "tech"))
inspect(tdm[idx, 1])
```

```
## <<TermDocumentMatrix (terms: 2, documents: 1)>>
## Non-/sparse entries: 0/2
## Sparsity           : 100%
## Maximal term length: 5
## Weighting          : term frequency (tf)
## Sample             :
##        Docs
## Terms   1
##   tesla 0
##   tech  0
```

We can see the most frequent word using the *findFreqTerms* which have lowest frequency of 20.

```
freq.terms <- findFreqTerms(tdm, lowfreq = 20)
```

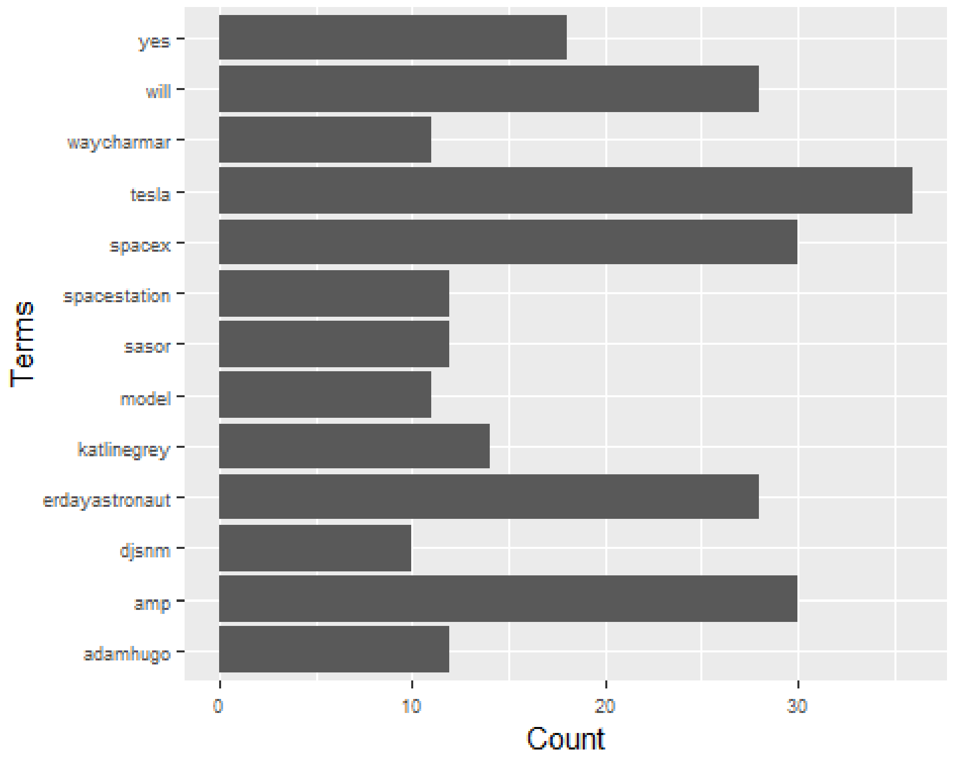
We have a word in each column, if we row sum the columns we will get the frequency of each word in complete corpus. Words having frequency lower than 10 are not of our interest so will get rid of them, and then convert this into a dataframe.

```
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq >= 10)
df <- data.frame(term = names(term.freq), freq = term.freq)
```

## 4) Freuncy Bar Graph

The best way to get see data is to visualize it with graph. Bar grapgh is used to see the frequency of the corresponding variable.

```
library(ggplot2)
windows()
ggplot(df, aes(x=term, y=freq)) + geom_bar(stat="identity") +
xlab("Terms") + ylab("Count") + coord_flip() +
theme(axis.text=element_text(size=7))
```

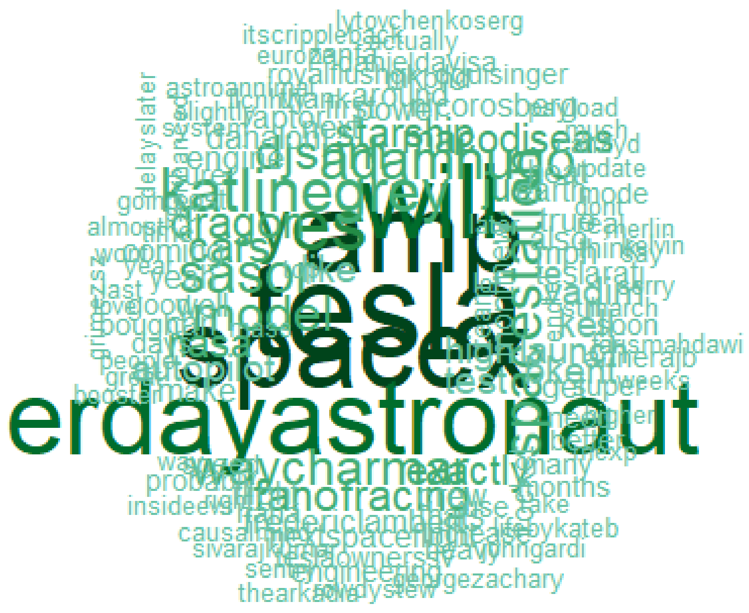


We can see leaders have very high frequency in our corpus.

## Word Cloud

Worldcloud is a a visible mass of condensed word varying size with there frequency. Firstly we will convert it into a matrix and then sort the word in a decerasing order of frequency and then plot the wordcloud.

```
m <- as.matrix(tdm)
word.freq <- sort(rowSums(m), decreasing = T)

pal <- brewer.pal(9, "BuGn")[-(1:4)]

windows()
wordcloud(words = names(word.freq), freq = word.freq,min.freq = 3,
random.order = F, colors = pal)
```



## 1) Assosiations

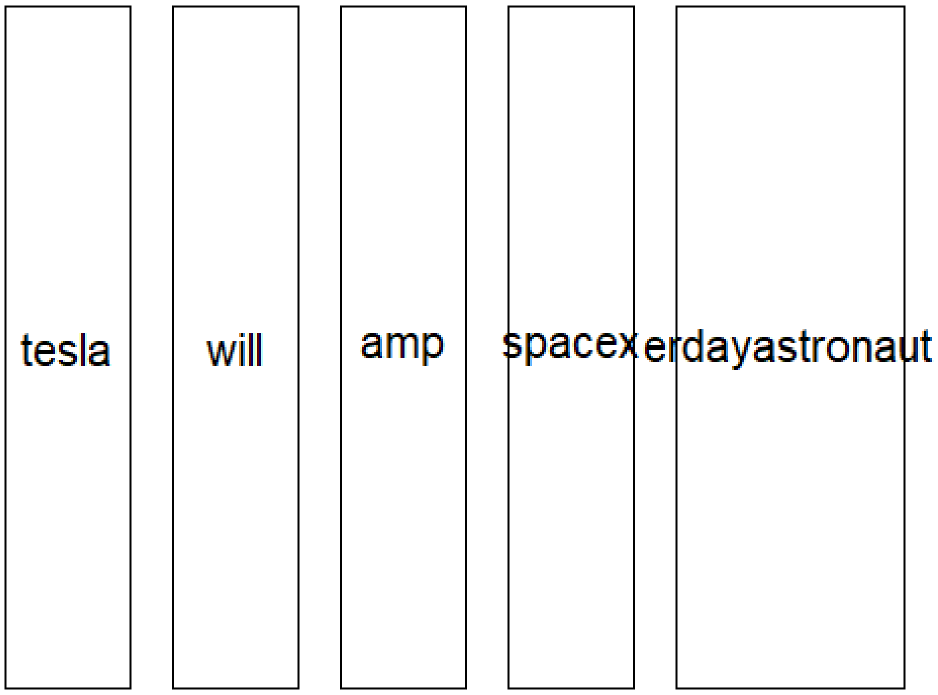
We can find assosiation between words using the *findAssocs* function. Pass the term document matrix as first, the word you want to find assosiation for as second and minimum assosiation as third argument.

```
findAssocs(tdm, "tesla", 0.2)

## $tesla
##             buy          comms         fixing        gfilche        mistake
##            0.33           0.32           0.32           0.32           0.32
##        customer           give        program          refer itscrippleback
##            0.32           0.32           0.32           0.32           0.25
##     lifebykateb    bryanmacksc          raulv         friend            car
##            0.25           0.21           0.21           0.21           0.21
##            want       referral
##            0.21           0.21

findAssocs(tdm, "spacex", 0.2)

## $spacex
##          behalf          thank          potus           team        control
##            0.37           0.35           0.35           0.24           0.23
##          docked     rogiermaas          texas alexthechemist           nasa
##            0.23           0.23           0.23           0.23           0.21
##          raptor
##            0.20
```

## 2) Assosiation Graph

We can also plot a graph that shows assosiation between the words, greater this assosiation thicker the link between them.

```
library(Rgraphviz)
windows()
plot(tdm, term = freq.terms, corThreshold = 0.5, weighting = F)
```



## Topic Modeling

Topic modeling is a type of statistical model for discovering the abstract “topics” that occur in a collection of documents. We can find hidden semantic structures from the tweets. For this purpose we will use R package called *topicmodels*

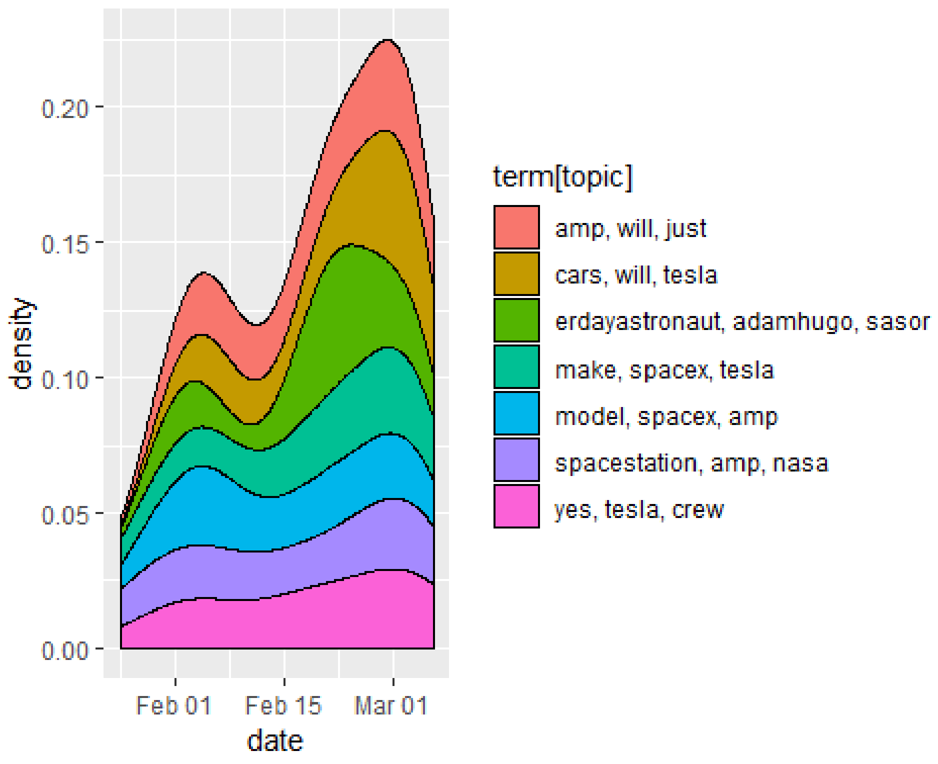
```
dtm <- as.DocumentTermMatrix(tdm)
library(topicmodels)
```

Each row of the input matrix needs to contain at least one non-zero entry

```
rowTotals <- apply(dtm , 1, sum)
dtm.new   <- dtm[rowTotals> 0, ]
```

Now we will pass this dtm object to *LDA()* and pass 7 to divide the dataset into total of 7 topics. In order to see the first few terms of the Topic we will call the *term()* function on lda as first argument and no of terms as second. To see the topic identified we will use *topic()* function, and then plot a density graph to visualize or topics.

```
lda <- LDA(dtm.new, k = 7)
term <- terms(lda, 3)
(term <- apply(term, MARGIN = 2, paste, collapse = ", "))

##                                Topic 1                                 Topic 2
##                   "cars, will, tesla"                     "yes, tesla, crew"
##                                Topic 3                                 Topic 4
##                  "model, spacex, amp"                  "make, spacex, tesla"
##                                Topic 5                                 Topic 6
##                     "amp, will, just" "erdayastronaut, adamhugo, sasor"
##                                Topic 7
##              "spacestation, amp, nasa"

topics <- topics(lda)

topics <- data.frame(date=as.IDate(tweets.df$created[c(1:200,1)]),
topic=topics[c(1:200,1)])
library(data.table)

windows()
ggplot(topics, aes(date, fill = term[topic])) +
geom_density(position = "stack")
```



## Semantic Analysis

Finally we will do semantic analysis on the tweets and see what are the semantics of the Elon Musk profile.

We will use *semantic* package in R for this purpose and pass the tweets to it. This package classifies the tweets into 3 sentiments negative,neutral or positive.

```
library(sentiment)
sentiments <- sentiment(tweets.df$text)
```

This returns us a dataframe with 3 attributes tweets, polarity and language, polarity is the attribute which represents the sentiment of the corresponding tweet. To display the total tweets in each sentiment we will use *table* function.

```
table(sentiments$polarity)

##
## negative  neutral positive
##        2      209       46
```

We can see from the above output that we have 2,211 and 48 negative, neutral and positive tweets repectively.

Now we will introduce a new attribute in sentiments dataframe *score* and will map positive tweets to 1, negative tweets to -1, and neutral to 0.

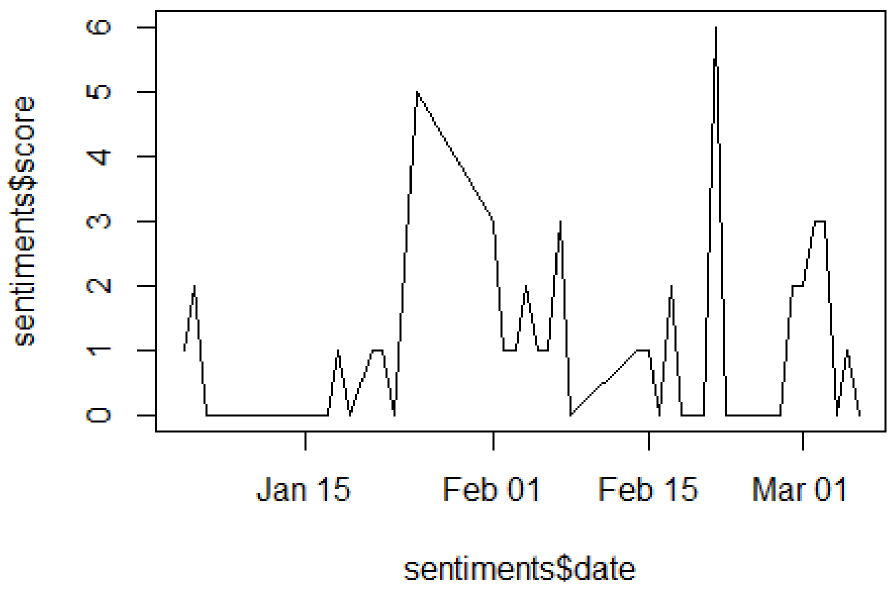
```
sentiments$score <- 0
sentiments$score[sentiments$polarity == "positive"] <- 1
sentiments$score[sentiments$polarity == "negative"] <- -1
```

We will create another variable date to store the creation date of the tweets.

```
sentiments$date <- as.IDate(tweets.df$created)
```

Now we will plot the graph and visualize how the sentiments of the people have changed over time. On the x-axis we have date and the y-axis we have the polarity of the tweet.

```
result <- aggregate(sentiments$score ~ sentiments$date, data = sentiments,
sum)
windows()
plot(result, type = "l")
```



## Conclusion

After processing the tweets of technology entrepreneur and engineer Elon Musk, We can see that most of the tweets of Elon Musk have a neutral sentiment, almost none of the tweets have negative sentiment.